Vysoká škola báňská – Technická univerzita Ostrava

Hornicko-geologická fakulta

Institut ekonomiky a systémů řízení

NÁVRH DATABÁZE RFID TAGŮ

Design of RFID Tags Database

Bakalářská práce

Autor: Stanislav Volný

Vedoucí práce: Ing. Pavel Staša, Ph.D.

OSTRAVA 2014

VŠB - Technická univerzita Ostrava
Hornicko-geologická fakulta
Institut ekonomiky a systémů řízení

Zadání bakalářské práce

Student: **Stanislav Volný**

Studijní program: B2102 Nerostné suroviny

Studijní obor: 6209R013 Informační a systémový management

Téma: Návrh databáze RFID tagů
Design of RFID Tags Database

Zásady pro vypracování:

Po analýze požadavků na funkční databázi RFID tagů v mezinárodní RFID laboratoři na VŠB-TUO se zabývejte návrhem takovéto databáze.

Osnova:
1. Úvod
2. Analýza požadavků
3. Návrh databáze RFID tagů
4. Závěr

Rozsah práce cca 25 stran textu

Seznam doporučené odborné literatury:

Seznam doporučené literatury:
[1]Pokorný, J.: Databázové systémy 2; Nakladatelství ČVUT; 1. vydání, Praha 2007; 190 s.; ISBN: 978-80-01-03797-3
[2]Hernandez, M., J.: Návrh databází; Grada Publishing, a.s., Praha 2006, 408 s., ISBN 80-247-0900-7

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Staša, Ph.D.**

Datum zadání: 31.10.2013
Datum odevzdání: 30.04.2014

doc. Dr. Ing. Oldřich Kodym
vedoucí institutu

prof. Ing. Vladimír Slivka, CSc., dr.h.c.
děkan fakulty

2014

PROHLÁŠENÍ

- Celou bakalářskou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.

- Byl jsem seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo.

- Beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3).

- Souhlasím s tím, že jeden výtisk bakalářské práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že údaje o bakalářské práci, obsažené v Záznamu o závěrečné práci, umístěném v příloze mé bakalářské práce, budou zveřejněny v informačním systému VŠB-TUO.

- Souhlasím s tím, že bakalářská práce je licencována pod Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported licencí. Pro zobrazení kopie této licence, je možno navštívit http://creativecommons.org/licenses/by-nc-sa/3.0/

- Bylo sjednáno, že s VŠB-TUO, v případě zájmu o komerční využití z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.

- Bylo sjednáno, že užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu komerčnímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne …… ……..

Jméno a příjmení

2014

PODĚKOVÁNÍ

V první řadě chci poděkovat vedoucímu práce panu Ing. Pavlu Stašovi, Ph.D. za pomoc a odborné rady, které mi poskytl při vypracovávání práce. Dále chci poděkovat své rodině za neustálou podporu při studiu a tvorbě bakalářské práce.

2014

ANOTACE

Cílem této práce je návrh relačního databázového systému pro mezinárodní laboratoř VŠB-TUO. Bakalářská práce reaguje na potřeby pracovníků laboratoře mít na jednom místě uložené technické informace o RFID čipech. Teoretická část se zaměřuje na vysvětlení pojmu databázového systému, jeho modelování a abstrakci. Následně jsou popsány metodiky a fáze vývoje databázového systému. V praktické části práce je provedena analýza a sběr požadavků na výslednou databázi. Dále je znázorněn postup návrhu konceptuálního a logického modelu databáze. Poslední částí práce je návrh fyzické realizace databáze v aplikaci MS access.

Klíčová slova: databázový systém, návrh, RFID, RFID tag

ABSTRACT

The purpose of this work is to design a relational database system for international laboratory VŠB-TUO. Bachelor thesis responds to the needs of laboratory personnel have in one place stored technical information about RFID chips. The theoretical part focuses on the notion of a database system, its modeling and abstraction. Subsequently there are described the methodology and database system development phases. In the practical part of the thesis are displayed gathering and analysis requirements on the resulting database. It is also shown the conceptual design process and the logical database model. The last part is the design of the physical implementation of the database in MS Access.

Key words: database system, design, RFID, RFID chip

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

DS - Databázový systém

SŘBD – Systém řízení báze dat

ERD – Entitně relační diagram

DFD – Data flow diagram

RFID – Rádio frekvenční identifikace (Radio frequency identification)

1. ÚVOD

V dnešní době informačních a komunikačních technologií, sociálních sítí a dynamicky tvořených webových stránek stoupá množství přenesených dat po síti. Proto narůstá potřeba najít způsob jak tyto informace efektivně uchovávat a organizovat. Pro tento účel vznikly databázové a informační systémy. Oba prostředky jsou umělé systémy vytvořené člověkem sloužící ke sběru, přenosu, zpracování a uchovávání dat.

Členové RFID laboratoře na VŠB – Technické univerzitě Ostrava dnes a denně pracují s velkým množstvím RFID tagů. Laboratoř disponuje několika stovkami RFID tagů od různých výrobců, různého provedení a určené pro široké spektrum průmyslového odvětví.

Motivem k vypracování uvedené bakalářské práce bylo navrhnout databázový systém, který má za cíl usnadnit zaměstnancům laboratoře vyhledávání informací o technických parametrech RFID čipů a díky tomu i samotnou manipulaci s těmito čipy. V současné době nemá laboratoř žádnou fungující databázi, veškeré informace o čipech jsou uvedeny v papírové kartotéce, která je nepřehledně uspořádaná.

Výsledné finální řešení v podobě plně funkční databáze není hlavním cílem této bakalářské práce, nicméně na základě této bakalářské práce a výsledného návrhu řešení je do budoucna možné připravit nástroj, který pracovníkům laboratoře usnadní a urychlí čas strávený s vyhledávání RFID tagů a informacemi potřebnými k testování RFID technologie.

2. DATABÁZOVÝ SYSTÉM

Z obecné teorie systémů lze říci, že systém je uspořádaná množina prvků a vazeb mezi nimi s určitým chováním a vykazující určité vlastnosti. Z výše uvedené definice lze stanovit i definici pro databázový systém:

Databázový systém je množina dat (báze dat) a programového vybavení (SŘBD) zpravující tuto množinu dat.

DBS=SŘBD+DB

Společně vykazují následující vlastnosti:

- „Struktury datových souborů jsou odděleny od aplikačních (uživatelských) programů.
- Přístup k datům je možný jen prostřednictvím programů databázového systému.
- Data je možné vyhodnotit jakýmkoliv způsobem.
- Je umožněn přístup více uživatelů současně a vyřešena ochrana dat před zneužitím.“ [1]

Data jsou údaje získané měřením, pozorováním nebo zaznamenáním z reálné skutečnosti. Interpretací těchto dat a vztahů mezi nimi získáme informace.

2.1 SYSTÉM ŘÍZENÍ BÁZE DAT

Systém řízení báze dat, neboli zkráceně SŘBD, je programové vybavení umožňující definování datových struktur a datových souborů, řešící fyzické uložení dat ve vnější paměti počítače, umožňující manipulaci s daty a formátování vstupních i výstupních informací. Řídicí systém databáze, který sídlí mezi vlastní fyzickou vrstvou (daty) a uživatelem. Díky této vrstvě nemusí uživatel při práci s databází vědět naprosto nic o její skutečné fyzické podobě a způsobu, jakým jsou data uložena a udržována. [2,3]

2.1.1 JAZYKOVÉ PROSTŘEDKY SŘBD

Systém řízení báze dat používá tři typy jazyků pro práci s daty:

1. DDL (Data Definition Language) – skupina příkazů sloužící pro definici datových struktur databáze. Ať už se jedná o tabulky, primární a fiktivní klíče nebo o datový typ atributů. Jednotlivé příklady pro definici jsou: CREATE - pomocí tohoto příkazu jak už sám název napovídá lze vytvořit databázovou tabulku. Příkaz ALTER změní strukturu tabulky podle zadaných parametrů. DROP příkazem lze odstranit tabulky, databáze i celé objekty.
2. DML (Data Manipulation Language) – Soubor prostředků pro manipulaci s daty. Používá se pro aktualizaci dat v databázi (přidání, odstranění a změně dat), k výběru dat podle zadaných kritérií. K výběru dat slouží dotazovací jazyk (tzv. querry language). Jedná se o příkazy SELECT, DELETE, INSERT, UPDATE.
3. DCL (Data Control Language)- soubor příkazů pro řízení dat sloužící k nastavení, změnám a odebírání přístupových práv jednotlivých tabulek. Jedná se o příkazy GRANT, REVOKE, CREATE USER, ALTER USER, DROP USER. K těmto příkazům je ještě přidružena podskupina pro řízení transakcí (Transaction control commands): COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION.
   [3,4]

2.1.2 TRANSAKČNÍ ZPRACOVÁNÍ

Databázovou transakci můžeme chápat jako posloupnost operací a příkazů, které převedou bázi dat z jednoho konzistentního stavu do druhého. Transakce začíná příkazem START a končí příkazem COMMIT, příkaz ROLLBACK celou transakci při jejím neúspěšném průběhu přeruší. Transakce musí proběhnout jako celek. V případě neúspěchu se veškeré změny ve struktuře databáze, které transakce provedla, musí navrátit do původního stavu. Transakce musí vykazovat následující vlastnosti: [3,4]

- Atomárnost (Atomicity) - Transakce je na dané rozlišovací úrovni dále nedělitelná. Vykoná se buď jako celek, nebo se nevykoná vůbec.
- Konzistence (Consistency) - Po dokončení transakce nesmí být porušena žádná integritní omezení. Data musí zůstat nadále konzistentní.

- Izolovanost (Isolation) - Operace které proběhnou uvnitř transakce, jsou skryty před uživateli. Důsledkem tohoto skrývání transakcí je nemožnost ovlivnění ostatních transakcí při chybném průběhu transakce.
- Trvalost (Durability) - Výsledek transakce musí být trvalý. Změny, které se provedou s daty, se fyzicky zapíší do databáze a již nemohou být ztraceny. [4,5]

2.2 OPERAČNÍ DATABÁZE

Operační databáze se používají v online zpracování transakcí OLTP tj. v situacích kde se uchovávají dynamicky měnící se data každý den. Jsou určeny a optimalizovány pro zpracování řady operací vykonávaných při provádění každodenních činností podniku (vytvoření faktury, příjem materiálu, vytvoření dodacího listu). [7]

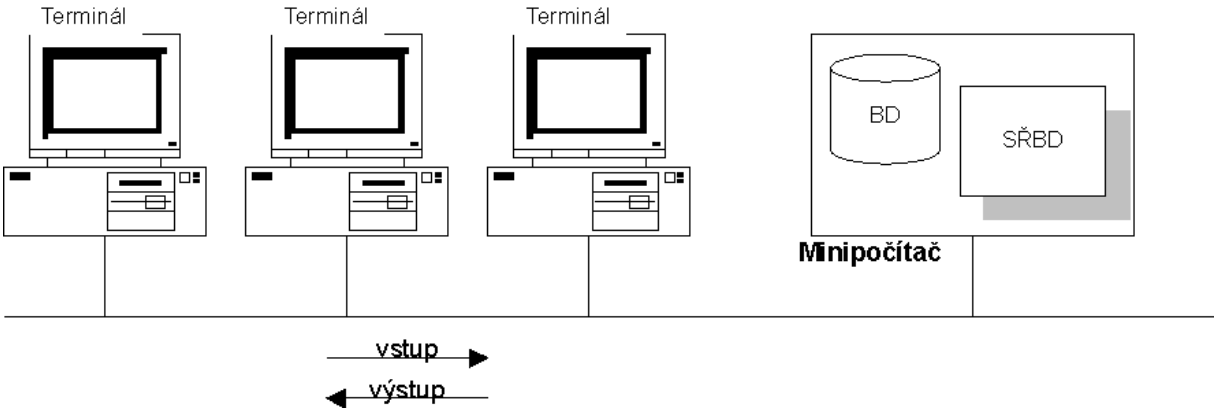
2.3 ANALYTICKÉ DATABÁZE

Analytické databáze se používají pro online analytické zpracování dat známé také pod zkratkou OLAP. Při tomto zpracovávání je potřeba uchovávat velké množství neměnných a časově závislých dat. Tyto data se používají pro sledování trendů, zobrazování statistických ukazatelů za dlouhé časové období. Na základě těchto informací se provádí rozhodnutí na taktické nebo strategické úrovni podniku. Analytické databáze získávají data hlavně z operačních databází. [2]

2.4 ARCHITEKTURA DATABÁZOVÝCH SYSTÉMŮ

2.4.1 CENTRÁLNÍ ARCHITEKTURA

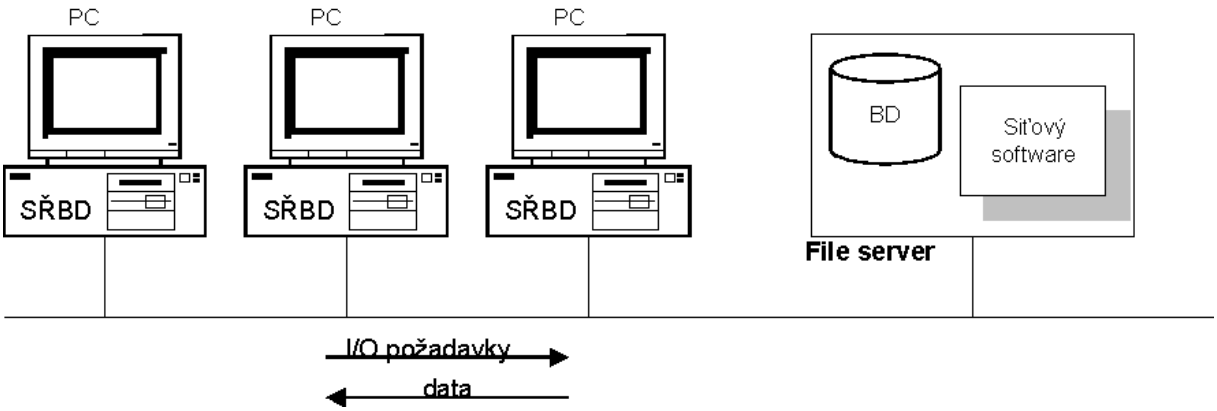
U centrální architektury datová báze i systém řízení báze dat běží na centrálním počítači. Uživatelé pracují na jednotlivých terminálech, které přenášejí data po síti. Po příchodu dat do centrálního počítače se zpracují programovým prostředkem (SŘBD).[6]



Obrázek 1 ukázka centrální architektury DS; zdroj: [20]

2.4.2 ARCHITEKTURA FILE-SERVER

V architektuře file-server je systém řízení báze dat a databázová aplikace uložena na jednotlivých počítačích, zatímco data jsou uložena na file-serveru. Ke komunikaci mezi aplikaci a bází dat dochází při zadání dotazu, následně SŘBD přijme dotaz a zašle jej na server. File-server zašle bloky dat na lokální počítač, zde se data zpracují podle zadaného dotazu a výsledek dotazu se zobrazí v aplikaci na počítači. Tato metoda se používala zejména při rozmachu osobní sítě a osobních počítačů. [6]

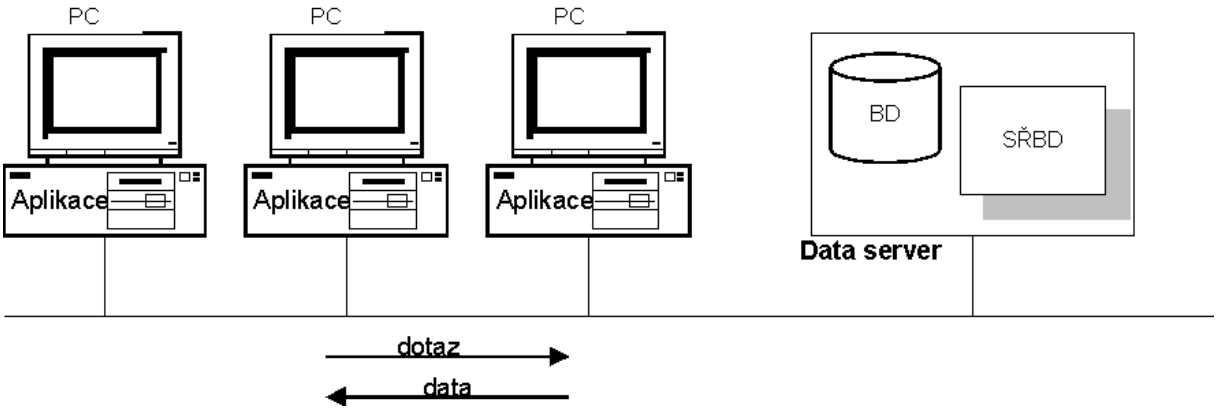


Obrázek 2 architektura file-server; zdroj: [20]

2.4.3 ARCHITEKTURA KLIENT-SERVER

Báze dat je uložena na počítači sloužícím jako databázový server. Uživatelé s daty pracují pomocí aplikace spuštěné na vlastních počítačích, takzvaných databázových klientech. Architektura klient-server implementuje integritu a konzistentnost dat na databázovém

serveru a tím umožní vytvořit množství uživatelských aplikací pracující nad jednou množinou dat, aniž by byla ovlivněna bezpečnost a integrita dat. [2]

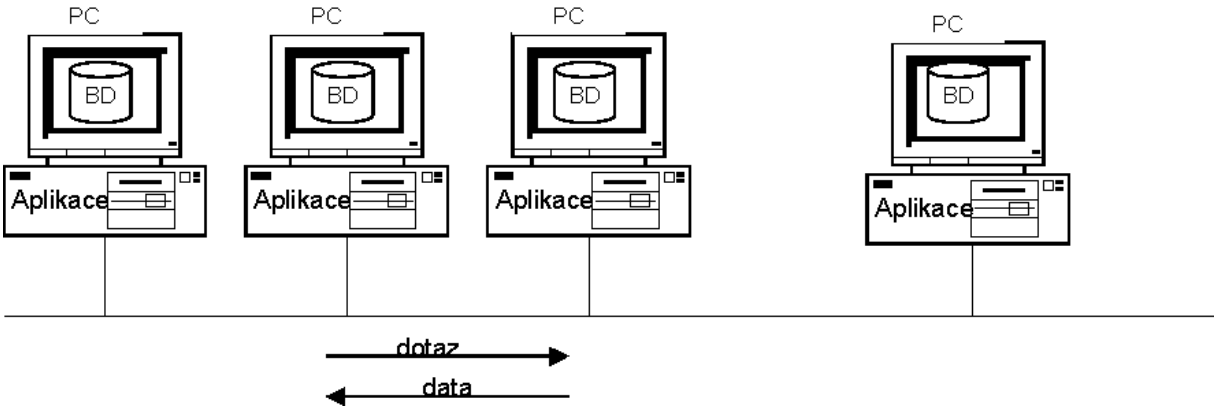


Obrázek 3 architektura klient-server; zdroj: [20]

2.4.4 DISTRIBUOVANÉ BÁZE

Pod pojmem distribuované databáze se označuje soustava osobních počítačů propojených sítí, kde každý z nich obsahuje svou vlastní bázi dat, SŘBD i aplikační vrstvu. Tyto počítače se navenek tváří jako jediný celistvý databázový systém. Distribuované báze vykazují tři vlastnosti:

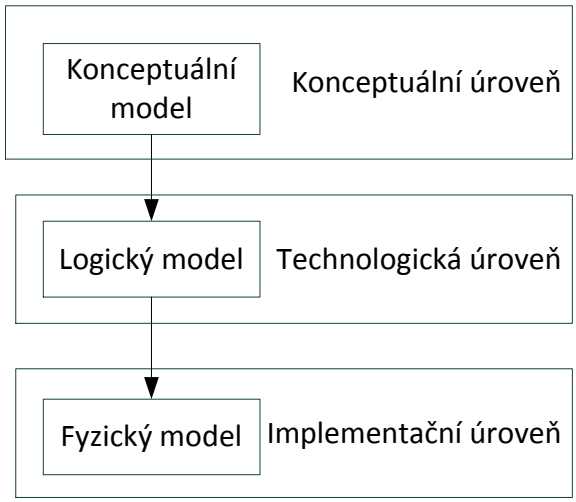
- Nezávislost na typu sítě - Pro propojení tohoto typu architektury je možné zvolit jakýkoliv druh sítě.
- Autonomnost - S každou částí báze dat je možné pracovat samostatně. Data jsou propojována dynamicky podle potřeby.
- Transparentnost - O místo uložení dat se stará SŘBD. [6]



Obrázek 4 architektura distribuovaných databázových systémů; zdroj: [20]

3. ABSTRAKCE MODELOVANÉ REALITY

K abstrakci modelovaného systému se používá principu tří architektur, známý také pod zkratkou P3A. Princip tři architektur umožňuje rozčlenit zkoumanou část reality na menší, lépe pochopitelné, části.



Obrázek 5: úrovně abstrakce modelované reality; zdroj: vlastní vypracování.

3.1 KONCEPTUÁLNÍ ÚROVEŇ

Cílem konceptuální úrovně je vytvořit konceptuální model, tj. popis objektů zájmové reality a vztahů mezi nimi, o kterých bude informační systém shromažďovat data. Při tvorbě konceptuálního modelu nesmí být brán ohled na fyzické uložení dat v databázi a na pozdější implementaci, jde tedy o nejvyšší míru abstrakce modelovaného systému. Konceptuální model určuje, co bude obsahem systému.

Konceptuální model může být vyjádřen dvěma způsoby. Jeden ze způsobů je tzv. lineární zápis, který popisuje entity a vztahy textem.

ENTITA_1 (atribut_1, atribut_2)

ENTITA_2 (atribut_1, atribut_2)

VZTAH (ENTITA_1, ENTITA_2, vztahový_atribut_1, vztahový_atribut_2). [10]

Druhým způsobem vyjádření konceptuálního modelu je Entitně-Relační diagram (ERD). Tento způsob znázornění je přehlednější a častěji využíván, neboť se jedná o grafické zaznamenávání entit a vztahů. V E-R diagramu se využívá následující grafické prvky:

- Obdélník - vyjadřující objekt zájmové reality- entitu. V záhlaví obdélníku je uveden název entity.
- Kosočtverec - vyjadřující vztah mezi dvěma entitami.
- Ovál - Pomocí oválu se znázorňují atributy jednotlivé entity.

3.2 TECHNOLOGICKÁ ÚROVEŇ

Při tvorbě technologické úrovně je vytvořen logický datový model, který zobrazuje veškeré objekty navrhovaného systému, kde u každého objektu jsou zobrazeny veškeré atributy spolu s primárními a fiktivními klíči (jeden řádek tabulky odpovídá jednomu atributu), a integritní omezení (jednotlivé sloupce přiřazují každému atributu integritní omezení). Tento model přináší zobecnění, pomocí kterého získá datová základna nezávislost na konkrétním databázovém prostředku. Logický model stále nesmí být omezen způsobem implementace. Technologická úroveň určuje, jak bude obsah systému realizován. [11]

3.3 IMPLEMENTAČNÍ ÚROVEŇ

Implementační úroveň určuje, jak bude řešení navržené na technologické úrovni realizováno. Určí se, na jaké databázové platformě systém poběží, dále se určí programovací jazyk, vývojové a uživatelské prostředí. Implementační úroveň tedy definuje fyzické uložení dat na nejnižší úrovni a jejich reprezentaci. [11]

4. DATABÁZOVÉ MODELOVÁNÍ

4.1 DATOVÉ MODELOVÁNÍ

Datová analýza zkoumá objekty reálného světa z hlediska jejich vlastností a vztahy mezi nimi a okolím. Datová analýza se snaží zachytit statický pohled na jednotlivé objekty zájmové reality a jejich vzájemné propojení. Výsledkem datové analýzy je konceptuální model systému, který nezobrazuje přeměnu datových toků, ale jejich statické uspořádání. Součástí konceptuálního modelu je Entitně-relační diagram, lineární zápis entit a vztahů, datový slovník.

4.2 FUNKČNÍ MODELOVÁNÍ

Funkční modelování zachycuje veškeré procesy, které probíhají uvnitř systému. Spolu s procesy zachycuje funkční analýza i externí a vnitřní entity komunikující s informačním systémem. Funkční analýza vychází ze zadání informačního systému, z požadavků zadanými uživatelem, z vstupů, výstupů systému atp. Funkční model obsahuje 2 úrovně:

1. Vnější pohled – je grafický náhled na strukturu a hierarchii procesů.

Prvním vnějším pohledem na procesy systému je tzv. diagram datových toků (Data Flow Diagram- DFD). DFD je grafické zobrazení funkčního modelu systému, vyjadřující jakým způsobem si procesy předávají data. Stejně jako ERD musí být jednoduchý a názorný pro uživatele. Při tvorbě DFD se využívá následujících prvků:

- **Proces -** soubor činností měnící vstupy na výstupy. Znázorňuje se pomocí kruhového uzlu. Každá proces má v DFD jednoznačně určené číslo a název. Číslování funkce je ve formátu x. y, kde x představuje číslo nadřízeného procesu a y představuje úroveň rozkladu.
- **Funkce -** je činnost v podniku, která přeměňuje vstupní data na výstupní.
- **Terminátor**- objekt, který není součástí popisovaného systému, ale jeho podstatného okolí. Tento objekt komunikuje s popisovaným systémem. Terminátor se objevuje pouze v kontextovém, protože pouze v kontextovém diagramu se zachytává komunikace systému s podstatným okolím. Terminátor se značí obdélníkem a musí být jednoznačně pojmenován.

- **Datový sklad- „**místo dočasného nebo trvalého uložení dat“ [8]. Konkrétní způsob uložení dat nemusí nutně představovat databázi, ale může se jednat také o kartotéku, textový soubor, kniha atd. Protože se DFD modeluje na konceptuální úrovni, není návrh ovlivněn způsobem implementace. Datový sklad se používá při jakékoli časové prodlevě přesunu dat. Pro každý datový sklad musí existovat alespoň jeden vstupní a jeden výstupní datový tok. Data z datového skladu musí procházet přes proces.
- **Datový tok- „**vyjadřuje tok dat nebo informací z jedné části systému do druhé, z okolí systému do systému nebo ze systému do jeho podstatného okolí“ [8]. Datové toky tedy představují data, které vstupují do systému jako např. textové znaky, čísla, záznamy, uvnitř systému jsou zpracovávána jednotlivými procesy, a následně vystupují do podstatného okolí systému. Datový tok se znázorňuje orientovanou hranou (úsečkou). Datové toky jsou jednou z možností propojení procesů a terminátorů.

2. Vnitřní pohled- podrobnější vysvětlení jednotlivých procesů. Příkladem vnitřního pohledu na vyvíjený systém je *minispecifikace.*

Minispecifikace je popisem procesu na nejnižší úrovni rozkladu. Popisuje podrobně chování tohoto procesu, a jak jsou vstupní data přeměňována na výstupní. Minispecifikace se provádí pro každý proces na nejnižší úrovni.[8]

4.3 ČASOVÁ ANALÝZA

Časová analýza slouží k modelování chování systémů v závislosti na čase. Popisuje časovou návaznost datových toků a za jakých podmínek se jednotlivé procesy spustí. Tyto podmínky je důležité zaznamenat, protože v systému nelze provádět kdykoliv a jakoukoliv funkci. Proto ke správnému běhu systému musí být přesně určena posloupnost procesů. K popisu časové analýzy se nejčastěji používá Stavový diagram – STD. [8]

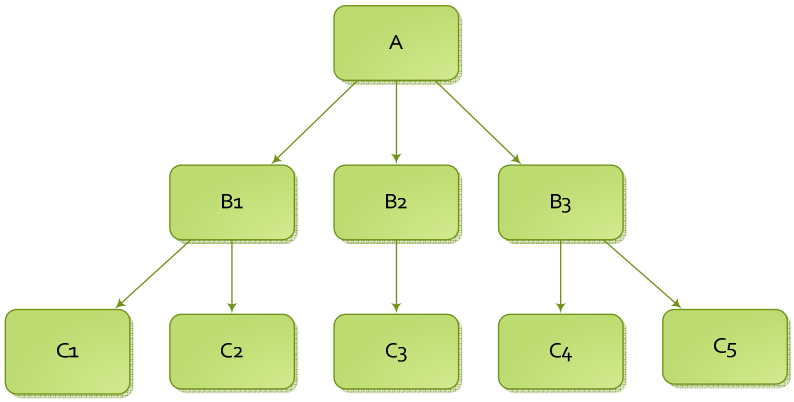
4.4 DATABÁZOVÉ MODELY

Databázový model je soubor pojmů sloužících k modelování objektové reality. Výsledkem datového modelování je databázové schéma, které popisuje strukturu uložených dat v databázi. [4]

4.4.1 HIERARCHICKÝ MODEL

Data v hierarchickém modelu jsou reprezentována hierarchicky. Uspořádání dat má obrácenou stromovou strukturu tzn., nejvýše postavená tabulka je nazývána kořen a ostatní tabulky z tohoto kořene vycházejí. Každý záznam představuje uzel ve stromové struktuře a má přesně dané předchůdce.

Vztahy mezi tabulkami jsou v hierarchické databázi reprezentovány pomocí pojmů – rodič a potomek. Kdy tabulka rodič může být spojena s jednou nebo více tabulkami potomků, ale tabulka potomka může být spojena pouze s jedinou tabulkou rodiče. Uživatel pak může k záznamům přistupovat ve směru hierarchie, tedy od kořenové tabulky, přičemž postupuje dále přes stromovou strukturu až ke hledaným. Proto při rušení jednotlivých záznamů mohou nastat problémy s existencí závislých záznamů.



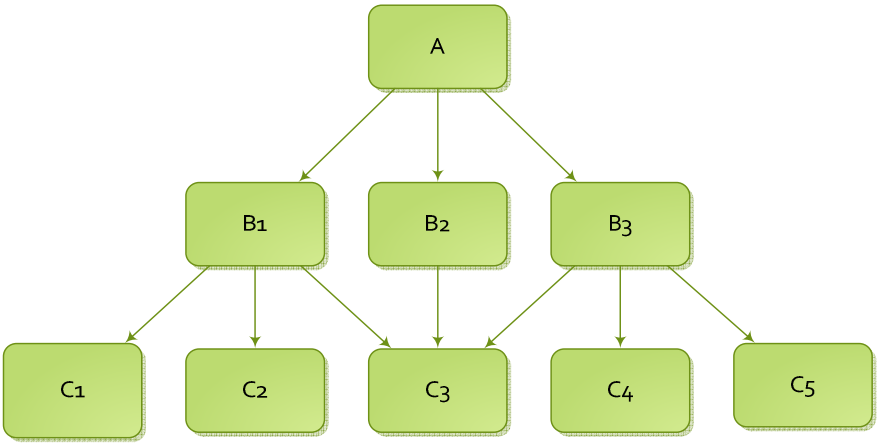
Obrázek 6 hierarchický model; zdroj: vlastní zpracování.

Díky přímému propojení tabulek je přístup k datům značně rychlejší než u ostatních typů databází. Výhodou propojení tabulek hierarchicky je automatické zajišťování integrity dat, protože existují pouze vztahy 1:N. Naopak nevýhodou hierarchické databáze je neexistence vztahů M:N. Další nevýhoda hierarchického uspořádání je složité vkládání a rušení záznamů. Při vkládání nových záznamů může mít za následek změny struktury dat. [12] [2],

4.4.2 SÍŤOVÝ MODEL

V roce 1971 byl skupinou DBTG (Data Base Task Group) ve sdružení CODASYL (Conference on Data System Languages) definován SŘBD síťového modelu. Tento model byl založen na souborech a vztazích mezi záznamy. Logický model databáze se nazývá schéma a má formu Bachmonova diagramu. Při definici schématu se typ entity nazývá typ záznamu-Record a obsahuje jména atributů čtyř typů - jednoduché, opakující se, složené nebo opakující

se složené. Jednotlivé záznamy s určitou kombinací hodnot odpovídajících položek se nazývají výskyty záznamů příslušného typu. Síťový model povoluje existenci redundance, takže se v databázi vyskytovat dva identické záznamy. Tyto záznamy jsou rozlišeny pouze hodnotou primárního klíče. [2]



Obrázek 7 síťový model; zdroj: vlastní zpracování

Síťový model definuje pouze funkcionální binární vztahy typu 1:1 a 1:N mezi dvěma typy záznamů. Tento vztah se nazývá set, C-množina nebo CS-typ. Sety propojují záznamy různého nebo stejného typu. Uzel reprezentuje soubor záznamů a množinová struktura reprezentuje a zřizuje vztah v síťové databázi. Výhodou síťové databáze je rychlý přístup k datům. Umožňuje uživatelům vytvářet dotazy, které jsou mnohem komplexnější než dotazy v hierarchickém modelu. [10]

Příklady operací prováděných nad databázovou strukturou:

- Vytvoř databázové schéma.
- Vytvoř nový záznam daného typu, zruš daný záznam, změň daný záznam.
- Vlož člen do výskytu setu daného vlastníka.
- Vyřaď člen z daného výskytu setu.
- Najdi první člen ve výskytu setu daného vlastníka.
- Najdi následníka ve výskytu setu daného vlastníka pro daný člen.
- Najdi vlastníka ve výskytu setu, známe-li určitý člen. [13]

4.4.3 OBJEKTOVÝ MODEL

Posledních 25 let se trend vývoje návrhu aplikací, ať jsou to desktopové nebo webové, změnil od strukturovaného k objektovému. Toto lze říci i o oblasti návrhu databází a informačních systémů. V osmdesátých letech 20. Století způsobili revoluci relační databáze, které nahradili síťové a hierarchické databáze. V letech devadesátých to pak byl objektový přístup, který změnil pohled na klasický návrh a modelování databází. V současnosti existují 2 typy objektových databází: [15].

- Objektově-relační - tento model jak už samotný název napovídá, vznikl sloučením relačního a objektového modelu. Jedná se o doplnění relačního modelu o možnost práce s datovými strukturami z objektově orientovaných programovacích jazyků.[5]
- Objektově orientovaný - „je založen na dekompozici informací z reálného světa na tzv. objekty.“[15]. Objektem je každá entita reálného světa, která je jednoznačně identifikovatelná. Objekt tak má unikátní identitu, i dva naprosto datově shodné objekty jsou vzájemně odlišné. Identita objektu je určena tzv. identifikátorem (označován jako OID), který je vytvořen systémem. Tento identifikátor je neměnný po celou dobu existence objektu a je skrytý jak pro programátora, tak i pro koncového uživatele. 1. Objektová databáze podporuje více typů množin objektů označovaných jako Sada (collection).[15,5]

Objekty jsou určeny pomocí tříd. Třída je abstraktním popisem objektu, určuje vlastnosti (datové složky) objektu a operace (metody), které lze s objektem provádět. Každý objekt je instancí určité třídy, počet těchto instancí je neomezený. Kromě objektů se v objektově orientovaném modelování používá pojem literál. Literál je entita datového typu, která nemá vlastní identitu. Literály se používají jako datové atributy objektů. S objekty se obvykle pracuje pomocí určité množiny základních operací. Důležitou operací prováděnou nad objekty je kopírování. Kopírování se rozlišuje na tzv. mělké (shallow) a hluboké (deep). Při mělkém kopírování dojde ke zkopírování všech atributů původního objektu, ale všechny vnitřní instance jsou navázány pouze referencí. Při hlubokém kopírování dojde ke zkopírování atributů kopírovaného objektu, ale také se vytvoří kopie objektů, na které odkazoval původní objekt. Při konstrukci objektu je volána funkce konstruktoru. Účelem konstruktoru je inicializovat objekt, konstruktor může být volán i několikrát po sobě. Součástí každé funkce je

tzv. destruktor. Destruktor je volán v momentě rušení objektu. Další operace prováděné s objekty jsou metody pro zjišťování a přiřazování hodnot atributů, metody pro výpočty a manipulaci s atributy, metody vytvářející výstupy pro uživatele.[15]

4.4.4 RELAČNÍ MODEL

Relační databázový model je založen na matematické teorii relačních množin a predikátové logice. V matematické teorii množin je relace chápána jako tabulka se sloupci a řádky. Každý sloupec relace představuje atribut a každý řádek relace představuje n-tici hodnot neboli záznamů. Relace je někdy nesprávně chápána jako vztah mezi dvěma tabulkami. Každý záznam v relaci je identifikován atributem, který obsahuje unikátní hodnotu. Toto jsou charakteristické vlastnosti relační databáze, které umožňují, že data mohou existovat bez závislosti na svém fyzickém uložení. Tímto se relační databáze odlišuje od hierarchické a síťové, kde znalost fyzického uložení dat je pro práci klíčová.[2]

Relační databázový systém je takový systém, který má následující vlastnosti:

1. databáze je uživatelem chápána jako množina relací a nic jiného.
2. Nad množinou dat se dají provádět minimálně operace selekce, projekce a spojení, aniž by existovaly jednoznačně předdefinované přístupové cesty pro uskutečnění těchto operací.

INTEGRITA RELAČNÍHO MODELU

integrita dat označuje konzistentnost, přesnost a platnost uložených dat v databázi. Jedná se o soubor pravidel, které dovolují vkládat data do databáze podle předem určených pravidel. Tyto pravidla určují například datový typ, velikost pole, vstupní formát dat atp.

Existují čtyři druhy integritního omezení, které se používají při návrhu databázového systému. První tři typy jsou založeny na struktuře uložení dat v databázi a jsou označeny podle oblasti, ve které se používají. Poslední typ je založen na způsobu jakým společnost používá své data. [2]

Entitní integrita - zamezuje výskytu duplicitního záznamu v tabulce a pole, které identifikuje tento záznam je jedinečné a neobsahuje nulovou hodnotu.

Doménová integrita - zabezpečuje strukturu každého pole v tabulce. Toto pole je spolehlivé a hodnoty jsou platné a konzistentní.

Referenční integrita - *„JE TAKOVÉ OMEZENÍ, KTERÉ ZAJIŠŤUJE RELACI MEZI TABULKAMI V RELAČNÍ DATABÁZI“* [3]. Záznamy ve vzájemně propojených tabulkách jsou v souladu. Pokud dojde ke změně dat v jedné tabulce, pak musí dojít ke změně souvisejících dat v druhé tabulce.

Business pravidla - zavádějí omezení na určité aspekty databáze založené na způsobu jakým podnik získává a zpracovává informace. Tato omezení mohou ovlivnit některé aspekty návrhu databáze, jako např. rozsah a typ hodnot uložených v poli, typ vztahu a povinnost zapojení do vztahu. [2]

V roce 1969 položil doktor E. F. Codd základy relačním databázím, když sepsal 12 pravidel pro tvorbu databází:

1. Databázový systém musí spravovat všechna data pouze pomocí relačních operací.
2. Veškerá data musí být reprezentována na logické úrovni jako hodnoty v relačních tabulkách.
3. Každý údaj v databázi musí být logicky dosažitelný pomocí kombinace názvu tabulky, názvu sloupce a hodnoty primárního klíče.
4. Každá neznámá hodnota musí být dosažitelná prostřednictvím ostatních známých hodnot.
5. Popis celé databáze musí být na logické úrovni reprezentován také relačním způsobem jako tzv. katalog – tedy také jako tabulka.
6. Pro komunikaci se SŘBD musí existovat minimálně jeden počítačový jazyk, který musí umožňovat: definici dat (DDL), integritní omezení, manipulaci s daty (DML), práci s transakcemi a autorizační pravidla.
7. SŘBD musí poskytovat možnost práce s pohledy do databáze včetně aktualizace obsažených dat.
8. Všechny operace vkládající a vybírající data musejí pracovat s tabulkami jako s celky.
9. Výsledky operací nesmějí být ovlivněny změnami struktur tabulek a konkrétní implementací databázového systému.
10. Výsledky operací nesmějí být ovlivněny změnami v integritních omezeních, pokud nedošlo ke změně dat.

11. Výsledky operací nesmějí být ovlivněny konkrétním rozmístěním dat v distribuované databázi.
12. Žádný uživatel databáze nesmí obcházet ani narušovat rozhraní SŘBD.[17]

4.4.5 NORMALIZACE

Je označována jako souhrn pravidel, postupů a procesů k úpravě datové struktury tabulek v relační databázi. Normalizace se provádí s cílem zamezení redundance, rozložení složitých relací na dvourozměrné tabulky s atomickými hodnotami a zabránění aktualizačním anomáliím.

Normalizace vede ke vzniku tabulek, které lze snadno udržovat a efektivně se dotazovat na data v nich obsažená. Normalizovaný technologický model musí zachovat všechny závislosti původního konceptuálního schématu a pomocí relací se musíme dostat k původním datům.

1. Normální forma- multizávislost

„Relace je v první normální formě, pokud jsou všechny její atributy definovány nad skalárními obory hodnot (doménami)“[16]

Z této definice vyplývá že, relace musí obsahovat pouze atomické hodnoty, tzn. hodnoty, které jsou na dané rozlišovací úrovni dále nedělitelné.

2. Normální forma

„Relace je ve druhé normální formě, pokud je v první normální formě a navíc všechny její hodnoty jsou závislé na celém primárním klíči“. [16]

V případě, že primární klíč je složený z více atributů pak musí být neklíčové atributy závislé na celém primárním a ne jen na jeho části.

3. Normální forma

„Relace je ve třetí normální formě, pokud je ve druhé normální formě a navíc všechny její neklíčové atributy jsou vzájemně nezávislé“. [16]

Tato normální forma vyžaduje, aby v relaci neexistovala tranzitivní závislost. Za tranzitivní závislost se označuje situace, kdy existují alespoň 2 neklíčové atributy a 1 klíčový.

Jeden neklíčový atribut je závislý na druhém neklíčovém atributu a druhý neklíčový atribut je závislý na primárním klíči.

Boyce Coddova normální forma

Boyce-Coddova normální forma je variací třetí normální formy. Relace se nachází v Boyce-Coddově normální formě, jestliže pro každou netriviální závislost $x \rightarrow y$ platí, že x je nadmnožinou určitého klíče ze schématu R. K tomu aby byla porušena BCNF musí být splněno několik podmínek:

1. Relace musí mít více kandidátních klíčů
2. Minimálně dva kandidátní klíče musí být složené z více atributů
3. Některé složené kandidátní klíče musí mít společný atribut.[16]

4. Normální forma

„Relace je ve čtvrté normální fomě, je-li v Boyce-Coddově normální formě a pokud jsou všechny vícehodnotové závislosti zároveň funkčními závislostmi z kandidátních klíčů.“[16].

5. Normální forma

„pokud je relace_1 spojena s relací_2, relace_2 je spojena s relací_3 a relace_3 je spojena zpětně s relací_1, pak všechny tři entity musí být součástí stejného vektoru hodnot“. [16]

Relace je v páté normální formě, pokud je ve čtvrté normální formě a není možné do ní přidat další atribut tak, aby se relace vlivem skrytých závislostí rozpadla na několik dílčích relací. [16, 2]

5. FÁZE VÝVOJE DATABÁZOVÉHO SYSTÉMU

Pod pojmem životní cyklus databáze, nebo jakéhokoli informačního a počítačového systému, se rozumí události, které nastanou od prvotního uvědomění potřeby používat systém, přes proces návrhu a zprovoznění, až po ukončení činnosti. [3]

Při vývoji databázového systému se z pohledu vývojářů databázových systémů využívají dva typy přístupů k vývoji DS - úkolocentrický a hodnotocentrický. První zmiňovaný přístup je založen na základech projektového řízení. Vývoj DS probíhá podle předem stanoveného plánu, který se skládá z řady dílčích projektů. Tyto projekty mají přesně stanovené datum odevzdání. Následující činnost může začít teprve až po ukončení činnosti předchozí. Úkolocentrický přístup je vhodný u projektů s jasně definovaným cílem a přesně stanovenou koncepci řešení. Na začátku projektu je jasně definována hodnota, kterou DS zákazníkovi přinese. Výhodou tohoto přístupu je kontrola a přehled nad probíhajícími činnostmi a postupem práce. Nevýhod úkolocentrického přístupu je hned několik. Pokud zákazník bude chtít změnu funkcionality systému v průběhu řešení, pak se tato změna obtížně integruje. Při nedostatečné analýze a neurčité formulaci řešení vzniknou vážné komplikace s dodržením zadané funkcionality systému. Kvalita konečného systému je definována podle splnění specifikace a dodržením harmonogramu a rozpočtu. [18]

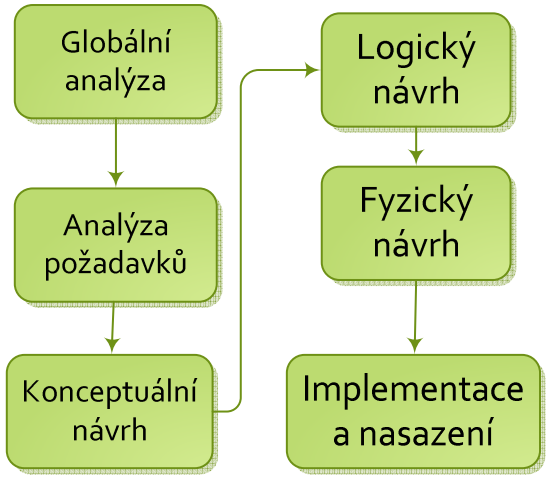
Hodnotocentrický přístup probíhá pravidelně v tzv. iteracích. Iterace se provádějí za přítomnosti zákazníka, ten může jakkoli ovlivnit proces návrhu. Při návrhu je tedy kladen důraz na hodnotu DS pro zákazníka. Harmonogram vývoje je také vytvořen, ale dodržení tohoto plánu není hlavním cílem. Části softwaru, které se v průběhu řešení ukážou jako zastaralé nebo nepotřebné, mohou být z konečného systému vyškrtnuty. Cílem je dodat zákazníkovi co nejdříve hotové části, tak aby mohl zhodnotit řešení a případně přidat další požadavky.

Hodnotocentrický přístup má:

- Zaměření na průběžné vytváření hodnoty.
- Zapojení zákazníka do řešení projektu.
- Očekáváme nejistotu a jsme na ni připraveni.
- Zdrojem hodnoty jsou jednotlivci – uznáním zvyšujeme kreativitu.
- Výkon povzbuzujeme skupinovou zodpovědností za výsledku a efektivitu týmu.

- Kvalita je definována přínosem pro zákazníka (termín může být posunut, pokud to zákazníkovi něco přinese).
- Změny a odchylky jsou brány jako přirozená součást řešení.
- Lze využít kreativity řešitelů.[18]

Na druhou stranu hodnotocentrický přístup je obtížněji řiditelný, obtížněji se vyhodnocuje postup práce. Existuje také velká pravděpodobnost, že projekt bude neúspěšně dokončen v řádném termínu. Tento postup nelze použít při projektů, kde investor a uživatel jsou odlišné subjekty.[18]



Obrázek 8 fáze vývoje DS; zdroj: vlastní tvorba

5.1 GLOBÁLNÍ ANALÝZA

Cílem této prvotní fáze je zpracování hrubého návrhu řešení databázového systému. Hrubý návrh zobrazuje fungování a strukturu systému, vazby mezi jednotlivými prvky systému a vztah systému k okolí. Dalším cílem je posouzení proveditelnosti (feasibility study) a výpočet návratnosti investice (ROI), která stanoví, zda li je u konečného produktu možné očekávat naplnění stanovených cílů. Tato studie také určí časovou a finanční náročnost. V neposlední řadě se specifikují rizika spojená s návrhem řešení.[18]

5.2 ANALÝZA POŽADAVKŮ

Během fáze sběru požadavků je shromažďuje a zdokumentován rámcový a dostatečně přesný popis cílů, kterých má projekt dosáhnout. Do seznamu požadavků je důležité zahrnout

co nejpřesnější informace o stávajících i budoucích obchodních i aplikačních procesech, aplikačních pravidlech a entitách. Hlavním cílem této etapy je rozpoznat modelovanou realitu a odpovědět tedy na otázku co bude obsahem systému.

Sběr požadavků lze provádět pomocí různých technik. Mezi nejčastěji používané patří rozhovory, dotazníky, pozorování a revize dokumentů. Nejvhodnější je používat kombinaci těchto metod pro získání nejpřesnějších informací a dat k sestavení požadavků. [3]

5.2.1 FUNKCIONÁLNÍ POŽADAVKY

Jsou požadavky na věcný a problémový obsah navrhovaného systému. Popisují tedy vstupy, výstupy systému, jaké funkce bude systém plnit, jaké je okolí systému, množství uživatelů přistupujících k aplikaci atp.

5.2.2 NEFUNKCIONÁLNÍ POŽADAVKY

Kromě věcné náplně systému je nutno zjistit další potřebné informace o okolnostech řešení. Nefunkční požadavky se definují až při návrhu, implementaci či uzavírání smlouvy. Nefunkční požadavky jsou několika druhů:

1. Požadavky na běh programu- Hardwarové nároky, spolehlivost, poměr výkon/cena, spolehlivost a jiné.
2. Požadavky na způsob řešení- použití metodik vývoje, programovací jazyk.
3. Vnější požadavky- finanční omezení, harmonogram, omezení daná legislativou, nutná integrace s jiným systémem. [2]

5.3 KONCEPTUÁLNÍ NÁVRH

O konceptuálním návrhu jsem se již zmiňoval v kapitole 2. Zde uvedu jen doplňující informace k této fázi vývoje. Mimo návrh E-R diagramu se v této fázi zhotoví tzv. externí neboli vnějšího návrhu. Součástí externího návrhu je vytvoření prototypu aplikace. Prototyp představuje náhled na finální aplikaci systému. Zobrazí se rozložení formulářů, tiskových sestav, webových stránek a další aplikace pro reprezentaci dat. Uživatelé tak získají představu, jak bude systém vypadat a jak se s ním bude pracovat. [2, 3]

5.4 LOGICKÝ NÁVRH

Tato fáze se v metodologiích nazývá interní návrh, protože se v této fázi navrhují interní (vnitřní) části systému, se kterými uživatel nikdy nepřijde do styku. Celá aplikace se rozdělí

do modulů, což jsou ucelené jednotky aplikačních programů. Tyto segmenty budou vytvářeny a testovány společně. Pro každý segment se vytvoří specifikace. Specifikace musí být dostatečně úplná, aby pomocí ní dokázal sestavit aplikaci jakýkoli programátor s dostačujícími dovednostmi. Logické toky dat mezi jednotlivými moduly se dokumentují pomocí diagramu toku dat. Hlavním těžištěm této etapy je normalizace. Tuto problematiku zmiňuji v kapitole 4. [2, 3]

5.5 FYZICKÝ NÁVRH

Při etapě fyzického návrhu se navržené a normalizované relace, vytvořené v logickém návrhu implementují do prostředí konkrétního SŘBD. To konkrétně znamená sestavení příkazy DDL jazyka. Tyto příkazy vytvoří databázové objekty, včetně klauzulí SQL definující fyzické uložení tabulek a indexů. Každé vytvořené tabulce se definuje primární klíč a každému atributu se definuje datový typ, velikost pole, a zdali jsou hodnoty pole povinné nebo ne.[2, 3]

5.6 IMPLEMENTACE A NASAZENÍ

Implementací se rozumí proces instalace nových komponent systému – samotnou databázovou aplikaci, webové stránky a webovou aplikaci, databázové objekty, sestavy, formuláře atp., na požadovaný hardware. Dále se databáze optimalizuje pro ideální běh na daném softwaru. Vytváří se také zabezpečení a zálohování databáze a nastaví se přístupová práva pro uživatele.[2, 3]

Nasazení pak znamená uvedení systému do provozu. Složitější a víceuživatelské aplikace mohou být při nasazování náročnější bezproblémový chod systému, proto se zavádějí do provozu postupně. Takto po určitý čas souběžně spolupracuje stará a nová verze aplikace, zatímco uživatelé jsou školeni a zaučováni pro práci s aplikací. [3]

6. PRAKTICKÁ ČÁST

V praktické části práce se budu zabývat analýzou požadavků a návrhem databáze pro RFID laboratoř Vysoké školy báňské – Technické univerzity Ostrava. Hlavní funkcí výsledné databáze bude rychlé, jednoduché a efektivní vyhledávání RFID čipů podle názvu, rozměrů, způsobu použití a frekvenčního pásma.

6.1 ILAB RFID

Mezinárodní laboratoř pro výzkum a vývoj RFID technologií (ILAB RFID) byla otevřena dne v roce 2009. Laboratoř vznikla ve spolupráci Vysoké školy báňské-Technické univerzity s oddělením průmyslového a systémového inženýrství na Dongguk univerzitě v Soulu. Laboratoř poskytuje služby po celém území ČR ať už pro aplikovaný výzkum, vývoj, testování nebo pro rozvoj řešení na bázi RFID technologií a standardů GS1 EPCglobal. Cílem tohoto projektu je propojení výzkumu RFID technologií se skupinou firem z praxe. Laboratoř je umístěna v centru pokročilých inovačních technologii v areálu VŠB a organizačně spadá pod institut ekonomiky a systémů řízení. [19]

Obrázek 9: logo laboratoře

6.2 ANALÝZA POŽADAVKŮ

Databáze bude uchovávat informace o RFID čipech se kterými pracovníci laboratoře každodenně manipulují. V současné době je databáze realizována ve formě šanonů a rozsáhlé kartotéky. Pracovníci musí při práci s čipy ztrácet mnoho času při procházení kartotéky, kde jsou jednotlivé čipy uloženy. Proto tato databáze má za cíl zefektivnit práci zaměstnancům laboratoře.

O každém čipu bude potřeba znát informace o typu, rozměrech, způsobu provedení, způsobu použití, informace o paměti, pracovní teplotě, hmotnosti a odolnosti. Dále v databázi bude uvedena evidence pracovníků laboratoře a dodavatelů čipů. U pracovníka je potřeba uchovávat informace o jménu, adrese, kontaktu na pracovníka. U dodavatele je potřeba znát název, sídlo firmy, kontakt.

6.2.1 FUNKCIONÁLNÍ POŽADAVKY

1. Evidence RFID čipů
2. Evidence pracovníků laboratoře
3. Evidence dodavatelů čipů.
4. Při vyhledávání bude možné filtrovat čipy podle názvu, způsobu použití, frekvenčního pásma a podle délky, šířky, výšky čipu.
5. Systém je před neoprávněním přístupem chráněn jménem a heslem. Uživatelská práva řešena nebyla.
6. Uživatel bude moci editovat a vyhledávat informace o čipech, dodavatelích a zaměstnancích.

6.2.2 NEFUNKCIONÁLNÍ POŽADAVKY

1. Návrh databáze je přizpůsoben pro aplikaci MS access 2010.
2. Databáze nebude součástí žádného informačního systému. Bude sloužit jako samostatná aplikace na jediném stolním počítači, ke kterému budou mít přístup všichni zaměstnanci.

6.3 KONCEPTUÁLNÍ NÁVRH

Po stanovení veškerých požadavků jsem sestavil konceptuální model sestávající se ze tří entit. Jedná se o entitu PRACOVNÍK, TAG a DODAVATEL. Jednotlivé atributy těchto entit vyplívají ze stanovených požadavků.

6.3.1 E-R DIAGRAM

Obrázek 10 konceptuální schéma

6.3.2 LINEÁRNÍ ZÁPIS ENTIT

PRACOVNÍK (*jméno, adresa, kontakt*)

TAG (*název, rozměr, použití, provedení, paměť, teplota, odolnost*)

DODAVATEL (*název, sídlo, kontakt*)

6.3.3 LINEÁRNÍ ZÁPIS VZTAHŮ

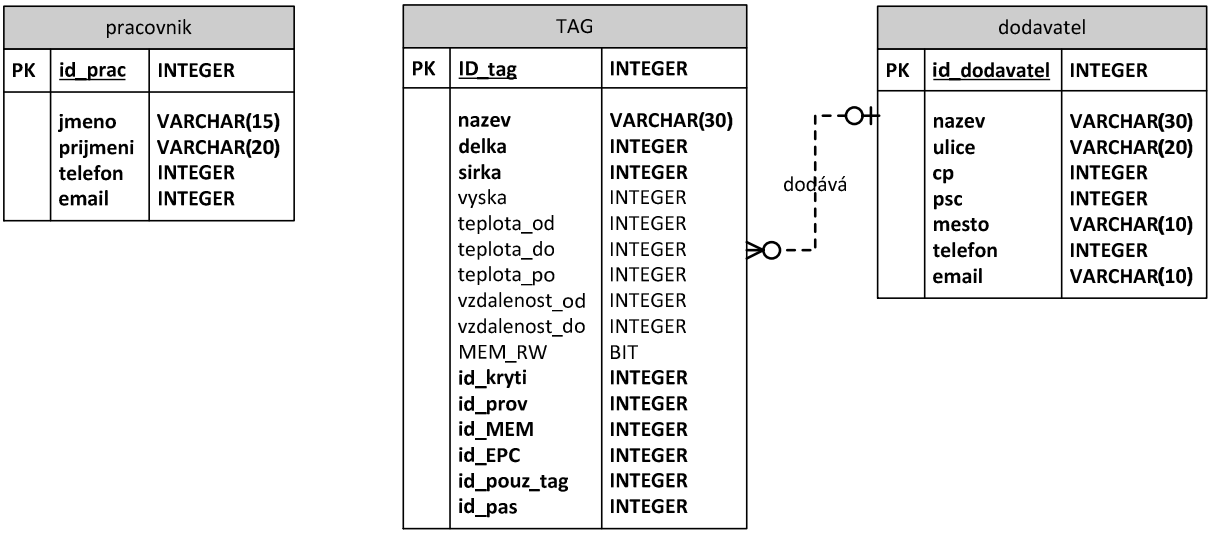
Dalším krokem konceptuálního návrhu bylo stanovení vztahů mezi entitami. Níže tedy uvedu výpis všech vztahů:

VYHLEDÁVÁ (*PRACOVNÍK, TAG*)

DODÁVÁ (*DODAVATEL, TAG*)

6.4 LOGICKÝ MODEL DAT

Nejdůležitějším krokem při tvorbě logického modelu dat je proces normalizace. Na diagramech č. 11 a 12 je znázorněno normalizované technologické schéma celé databáze. Pro lepší přehlednost jsem schéma rozdělil na dvě části. První část zobrazuje entity *tag*, *dodavatel* a *pracovník*.



Obrázek 11: Logický model části pracovník-tag-dodavatel

6.4.1 POPIS ENTIT, ATRIBUTŮ DATOVÝCH TYPŮ

Pracovník

Tabulka *pracovník* není propojena vztahem s další částí databáze, jelikož se jedná pouze o informativní tabulku. Tabulka bude obsahovat pouze informace o zaměstnancích laboratoře. Nad touto tabulkou budou prováděny dotazy k získání osobních údajů. Pole ID_prac je primární klíč, datový typ pole je INTEGER. Pole *jmeno, prijmeni, telefon, email* jsou povinná s hodnotou NOT NULL.

Dodavatel

Tabulka *dodavatel* slouží k evidenci dodavatelů tagů pro laboratoř. Kardinalita vztahu je 1:N. *ID_dodavatel* je primární klíč, datový typ pole je INTEGER. Pole *nazev, ulice, cp, psc, město, telefon, email* jsou povinná s hodnotou NOT NULL.

Tag

Slouží k evidenci RFID čipů. Jedná se o hlavní entitu celé databáze a je nadřazená tabulkám *dodavatel, pouziti_tagu, provedeni, IP krytí, velikost_MEM, MEM_EPC, pasmo.*

Tabulka 1: popis atributů entity tag

Název atributu	Vlastnosti
Id_tag	- primární klíč - Jednoznačný identifikátor čipu - Datový typ INTEGER
Nazev	- Datový typ VARCHAR s omezením na 30 znaků - NOT NULL - přesný název tagu
Delka	- Datový typ INTEGER - NOT NULL - délka tagu v mm
Sirka	- Datový typ INTEGER - NOT NULL - šířka tagu v mm
Vyska	- Datový typ INTEGER - NULL - výška tagu v mm
Teplota_od	- Datový typ INTEGER - NULL - maximální hranice pracovní teploty udávaná v °C
Teplota_do	- Datový typ INTEGER - NULL - minimální hranice pracovní teploty udávaná v °C

Teplota_po	-Datový typ INTEGER - NULL - doba vystavení tagu maximální teplotě udávané v °C
Vzdálenost_od	- Datový typ INTEGER - NULL - minimální čtecí vzdálenost tagu v m
Vzdálenost_do	- Datový typ INTEGER - NULL - maximální čtecí vzdálenost tagu v m
MEM_RW	- Datový typ bit (ANO/NE) - NULL -udává, zdali je paměť tagu přepisovatelná
Id_kryti	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce provedení
Id_prov	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce provedení
Id_MEM	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce velikost_MEM
Id_EPC	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce EPC_MEM
Id_pas	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce pasmo
Id_dodavatel	- datový typ INTEGER - NOT NULL - fiktivní klíč k tabulce dodavatel

Provedení

Značí způsob provedení jednotlivého tagu – zdali je tag zapouzdřen, na cívce, na plastovém proužku atp. Kardinalita vztahu je 1:N nadřazená tabulka je *tag*. Pokud existuje záznam v tabulce *tag* musí existovat záznam v tabulce *provedení*.

Pásmo

Značí, v jakém frekvenčním pásmu daný tag pracuje, zdali se jedná o UHF, HF nebo LF. Kardinalita vztahu mezi tabulkou *pásmo* a tabulkou *tag* je N:1. Parcialita vztahu je na straně tabulky *pásmo*.

Použití tagu

Vazební tabulka mezi tabulky *tag* a *použití*. Přiřazuje jeden záznam z tabulky *tag* k více záznamům v tabulce *použití*. Pokud existuje záznam v tabulce *tag,* musí existovat záznam v tabulce *použití_tagu*. Pokud existuje záznam v tabulce *použití_tagu*, musí existovat záznam v tabulce *použití*.

IP krytí

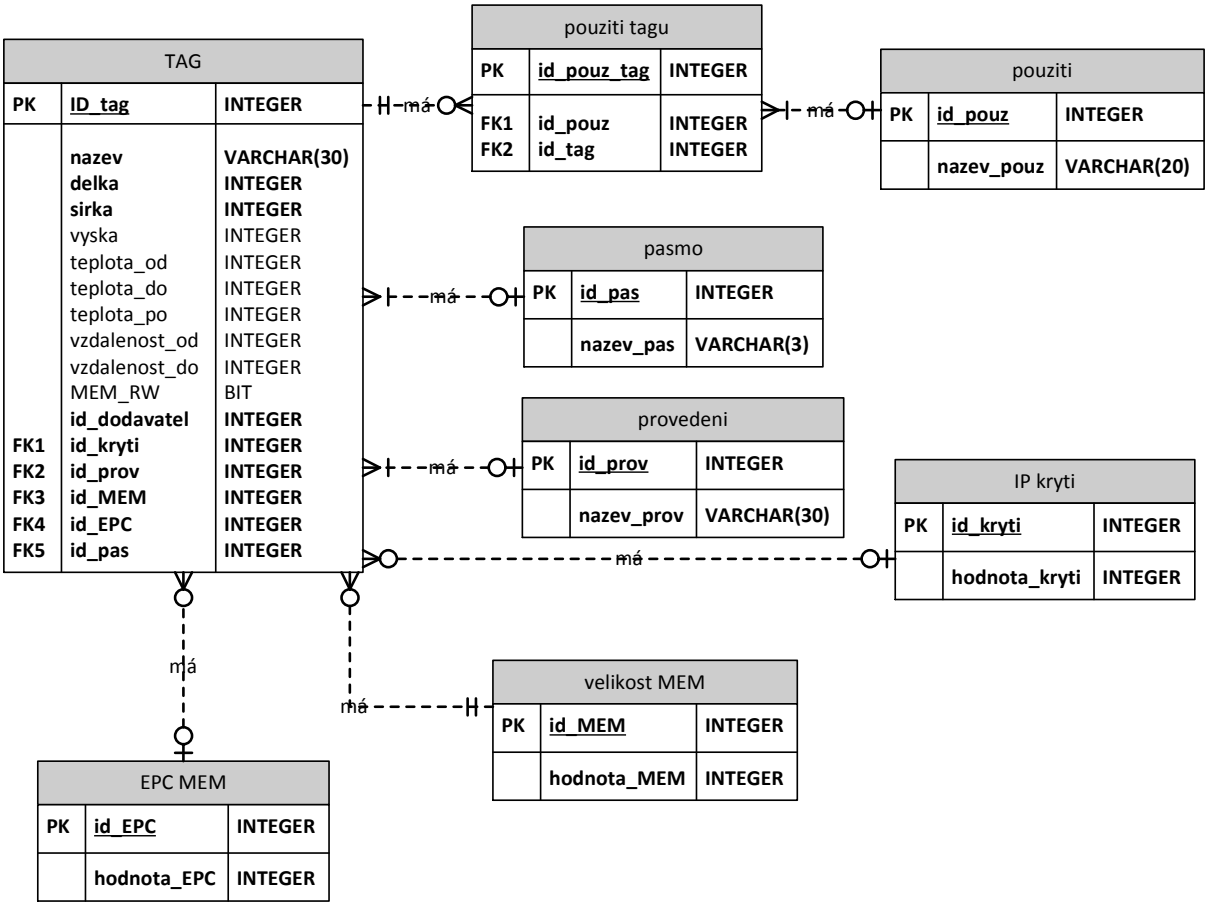
Určuje odolnost tagu vůči poškození. Udává se v číselných hodnotách, např. 67 značí odolnost proti pevným součástkám a vodě. Kardinalita vztahu je N:1. ID_krytí je primární klíč. *Hodnota_krytí* je povinný údaj proto pole nemůže zůstat prázdné- NOT NULL.

Velikost MEM

Značí kapacitu paměti jednotlivého tagu. Kapacita je udávána v bitech. Kardinalita vztahu je N:1. Pokud existuje záznam v tabulce *tag,* musí existovat záznam v tabulce *velikost_MEM*. Pokud existuje záznam v tabulce *velikost_MEM* nemusí existovat záznam v tabulce *tag*. *Hodnota_MEM* je povinný údaj- NOT NULL.

EPC MEM

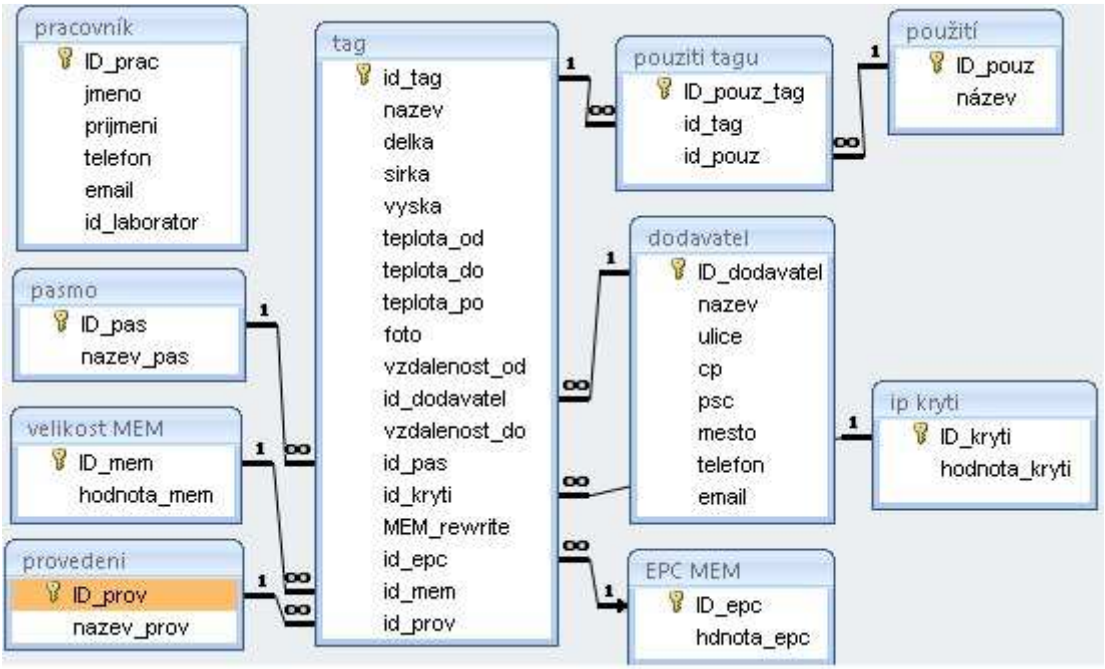
Značí kapacitu EPC paměti. Kapacita je udávána v bitech. Kardinalita vztahu je N:1. Pokud existuje záznam v tabulce *tag,* musí existovat záznam v tabulce *EPC_MEM*. *Hodnota_EPC* je povinný- NOT NULL



Obrázek 12: Logický model části TAG

6.5 FYZICKÝ NÁVRH

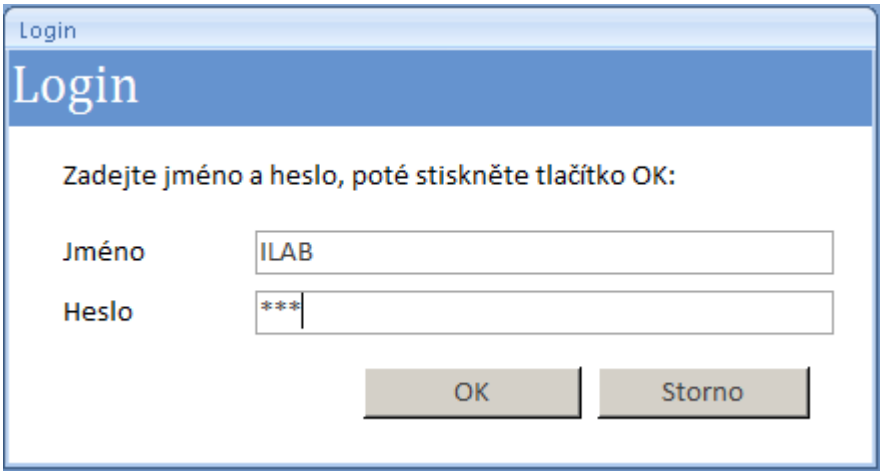
Posledním krokem při tvorbě databázového systému je návrh a realizace databázové aplikace spolu s návrhem uživatelského rozhraní, formulářů, dotazů a tiskových sestav. Přestože fyzická realizace nebyla hlavním cílem této práce, tím byl pouze návrh databáze, rozhodl jsem se po domluvě s vedoucím práce zhotovit prvotní verzi databáze vytvořenou v aplikaci MS Access 2007. Na obrázku 13 je znázorněné úplné relační schéma.



Obrázek 13: relační schéma v aplikaci access

6.5.1 ZABEZPEČENÍ APLIKACE

Problém bezpečnosti databázové aplikace bude řešen pomocí přihlašovací obrazovky zobrazené na obrázku 14. Uživatelé se budou moci do aplikace přihlásit pomocí jednotného uživatelského jména a hesla. Možnost přihlašovacích práv a uživatelských účtů není v této aplikaci řešeno.



Obrázek 14: přihlašovací obrazovka

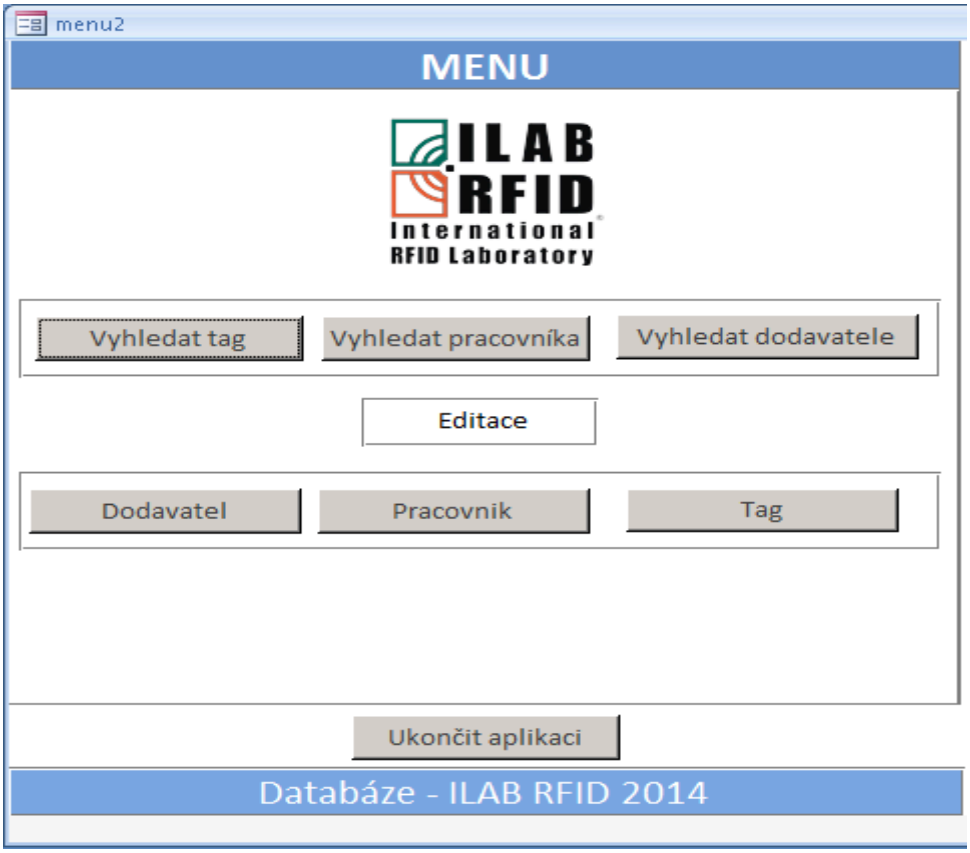
Níže uvedený skript zajistí kontrolu zadaného přihlašovacího jména a hesla. Pokud zadané údaje odpovídají předem definovaným znakům, zobrazí se uživateli zpráva „Přihlášení

proběhlo úspěšně. Vítejte.“ a otevře se formulář menu. Pokud zadané údaje neodpovídají, celá aplikace se ukončí.

```
Private Sub Tělo_Click()
jmeno.SetFocus
If jmeno = "ILAB" And heslo = "tag" Then
MsgBox "Přihlášení proběhlo úspěšně. Vítejte.", vbInformation, "ILAB RFID"
DoCmd.Close
DoCmd.OpenForm "menu"
Else
MsgBox "Nesprávně vyplněné údaje."
End If
End Sub
```

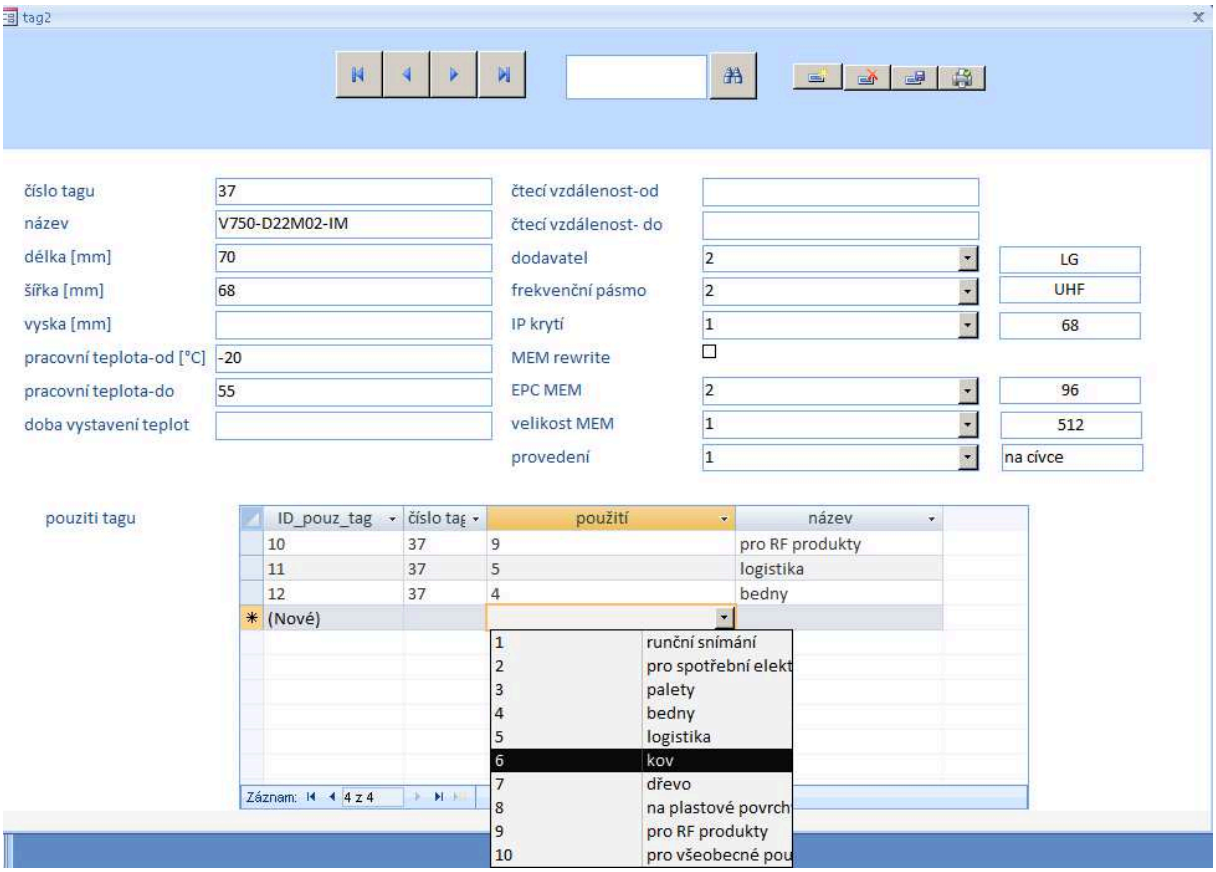
6.5.2 UŽIVATELSKÉ PROSTŘEDÍ

Na obrázku 15 je zobrazeno ovládací prostředí aplikace. Díky tomuto prostředí bude uživatel snadno ovládat celou aplikaci. Hlavní funkce aplikace jsou rozděleny na dvě části. Na vyhledávání, procházení záznamů ve formulářích, a na editaci informací o dodavatelích, pracovnících a čipech.



Obrázek 15: ovládací menu aplikace

obrázku 16 je zobrazen formulář pro vytváření nových záznamů. Tento formulář slouží k vytváření nových záznamů. Pomocí jednoduchých ovládacích prvků v horní části formuláře lze procházet a editovat již vytvořené záznamy. Další část formuláře se skládá s několika polí pro zadávání technických parametrů. Vnořený podformulář slouží k přiřazení způsobu použití k jednotlivým čipům.



Obrázek 16: formulář pro zadávání nových záznamů

7. ZÁVĚR

Cílem bakalářské práce bylo provedení analýzy požadavků na funkční databázi RFID tagů v mezinárodní RFID laboratoři na VŠB-TUO a následné navržení databázové struktury. Veškeré zadané cíle byly splněny.

V první části práce byl vysvětlen teoretický základ k problematice databázových systémů a jejich modelování a abstrakce. Spolu s problematikou modelování byly v práci popsány fáze vývoje databázových systémů a přístupy k vývoji databází.

V praktické části byla na základě rozhovoru vedeného s pracovníky laboratoře a prostudování současného stavu databáze, provedena analýza a sběr požadavků. Na základě informací zjištěných při analýze byl vytvořen konceptuální návrh, následně byla provedena normalizace a vytvořen logický model. V závěrečné části je také proveden návrh fyzické realizace databáze v aplikaci MS access a zobrazení uživatelského prostředí databáze. Tato fyzická realizace nebyla dokončena z důvodu časové náročnosti provedení a možného přesažení rozsahu práce. Do budoucna je mým plánem vytvořit databázi v prostředí MySQL s využitím programovacího jazyka php.

Hlavním přínosem práce je vytvoření implementačně nezávislých datových modelů, díky kterým lze fyzicky vytvořit databázový systém v kterémkoli vývojovém prostředí nebo programovacím jazyce. Hlavním přínosem pro svou osobu bylo využití teoretických znalostí s vývojem databázových systémů, nabytých v průběhu studia na vysoké škole.

8. POUŽITÁ LITERATURA

[1] ŠEDA, Miloš. *Databázové systémy*. Brno, 2002. Doplňující text ke konzultacím v 3. ročníku kombinovaného bakalářskéhúo studia oboru Aplikovaná infor. VUT FSI.

[2] HERNANDEZ, Michael. *Návrh databází*. druhé. Praha: Grada, 2006, 408 s. ISBN 80-247-0900-7.

[3] OPPEL, Andrew. *Databáze bez předchozích znalostí*. Brno: Computer Press, a.s., 2006. ISBN 80-251-1197-7.

[4] JAROSLAV, Pokorný a Ivan HALAŠKA. Databázové systémy. Vyd. 2. Praha: ČVUT, 2004, 148 s. ISBN 80-010-2789-9.

[5] Databázová transakce. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-03]. Dostupné z: <http://cs.wikipedia.org/wiki/Datab%C3%A1zov%C3%A1_transakce>

[6] Databázové systémy. *Homen* [online]. [cit. 2014-04-03]. Dostupné z: <http://homen.vsb.cz/~s1i95/ISVDAS/IS/IS_db_sys.htm>

[7] OLTP a OLAP systémy. *Forum-media* [online]. 2011 [cit. 2014-04-10]. Dostupné <z: http://www.forum-media.cz/res/data/004/000632.pdf>

[8] ŠARMANOVÁ, Jana. *DATABÁZOVÉ A INFORMAČNÍ SYSTÉMY.* Ostrava : VŠB – Technická univerzita Ostrava, 2007. ISBN 978-80-248-1499-5.

[9] KUČEROVÁ, Helena. Databázové systémy, sylaby ke kurzu.

[10] KYBKALO, Anatoliy. *Datové modelování*. Praha, 2013. Bakalářská práce. Unicorn College.

[11] Úvod do datového modelování. *Ing. Martin olhanec, CSc.: osobní stránky* [online]. 2007[cit.2014-04-10]. Dostupné z: <http://martin.feld.cvut.cz/~mmm/vyuka/X13DFA/files/UdDM.pdf>

[12] TELNAROVÁ, Zdeňka. *Úvod do databází I* [online]. Ostrava, 2003 [cit. 2014-04-10]. Dostupné z: <http://fakulty.osu.cz/prf/rsk/uploaded/1539_XUVDT1.pdf>. Distanční výuková opora. Ostravská univerzita.

[13] HRONEK, Jiří. *Databázové systémy* [online]. Olomouc, 2007 [cit. 2014-04-10]. Dostupné z: <http://phoenix.inf.upol.cz/esf/ucebni/databa.pdf>. Učební text. Univerzita palackého.

[14] Relační vs. objektově-relační vs. objektové databáze. [online]. [cit. 2014-04-10]. Dostupné z: <http://www.fi.muni.cz/~xbatko/oracle/compare.html>

[15] Objektové databáze. ŠVEC, Martin. [online]. 2003 [cit. 2014-04-10]. Dostupné z: <http://www.fit.vutbr.cz/study/courses/VPD/public/0203VPD-Svec.pdf>

[16] KOCH, Miloš. *Datové a funkční modelování*. Brno: Akademické nakladatelství CERM, s.r.o., 2004. ISBN 80-214-2724-8.

[17] Databázové systémy. *Izona.net* [online]. 2014 [cit. 2014-04-26]. Dostupné z: <http://alex.izona.net/vsczu/Treti%20semestr/Databazove%20systemy/Skripta/skripta.pdf>

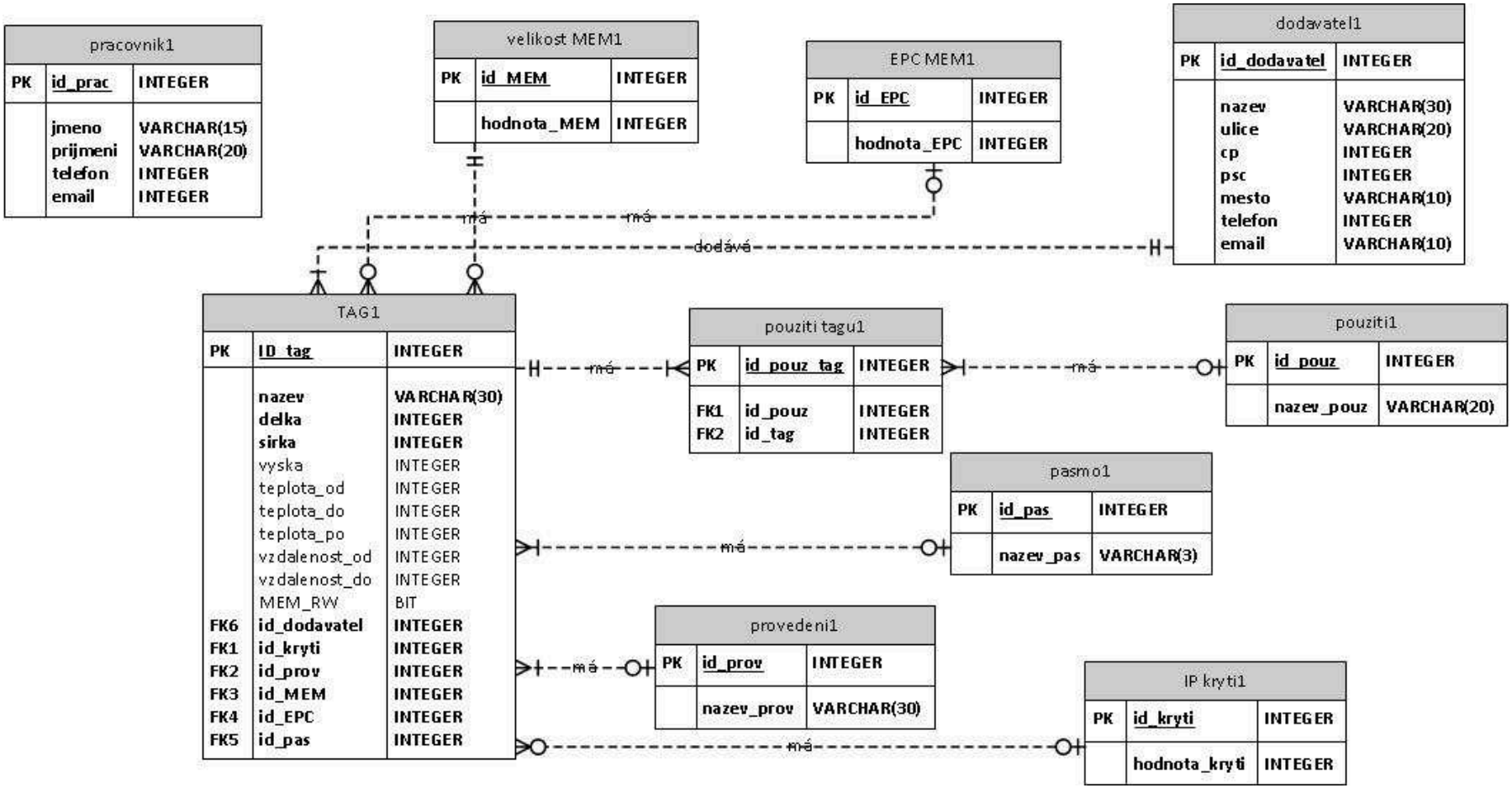
[18] Přístup k vývoji IS. DANEL, Roman. *Homel.vsb.cz/~dan11* [online]. 2013 [cit. 2014-04-26]. Dostupné z: <http://homel.vsb.cz/~dan11/is_skripta/IS%202011%20-%20Pristupy%20k%20vyvoji%20IS.pdf>

[19] Mezinárodní RFID laboratoř. [online]. 2014 [cit. 2014-04-26]. Dostupné z: <http://rfid.vsb.cz/cs/>m

[20] Index of /~s1i95/ISVDAS/IS. *Homen.vsb.cz* [online]. 2000 [cit. 2014-04-30]. Dostupné z: <http://homen.vsb.cz/~s1i95/ISVDAS/IS/>

9. SEZNAM OBRÁZKŮ A TABULEK

10. PŘÍLOHY



Obrázek 17: uplné logické schéma